Audio and QuickTime

Session 711

Audio and QuickTime

Tim Cherna
Engineering Manager
QuickTime A/V Foundation

Introduction

- Session covers changes to QuickTime audio foundation
- CoreAudio Yes, Sound Manager No
- Brings new audio capabilities to QuickTime
- Audio codecs replace sound codecs

What You Will Learn

- Playback
- Export
- DV Audio changes
- Audio codec news

Technology Framework

- QuickTime
- CoreAudio

Audio Playback

Greg Chapman
QuickTime Engineering

Lift the Restrictions Inherited from Sound Manager

- More than 2 channels per track
- More than 64 kHz sample rate
- More than 16 bits per integer sample
- Floating point samples
- Multitrack surround

Demo:
Audio Playback

How Did We Do That?

- New SoundDescription version
  (for instance, sample rate is Float64)
- New optional SoundDescriptionExtension
  (kAudioChannelLayout)
- New SoundMediaHandler (uses CoreAudio)
- New multitrack mixer

Historical Channel Mapping

- Movies can have one or more sound tracks
- Each track can have one or two channels
  - Two-channel tracks implied L R
- Multitrack movies always mix to mono or stereo

New Channel Mapping

- Tracks can have any number of channels
- The track's channels are assigned to speakers using a SoundDescriptionExtension
- Each track can contribute to the movie's channel layout. For example:
  - One 6-channel track
  - Six 1-channel tracks
  - A 5-channel track and a 1-channel LFE track

New Channel Mapping

- The track order is not important so long as the channel assignments are labeled
- Unlabeled mono and stereo tracks behave as before
- Multichannel movies are downmixed as necessary

Demo:
Channel Mapping

Interesting New APIs

- QTConvertSoundDescription
- SetMovieAudioOutputDeviceUID
- ExtractMovieAudio

Interesting New APIs

QTConvertSoundDescription

- Converts any version SoundDescription to any other version
- To “lowest possible version” is a useful option
- To/from AudioStreamBasicDescription

Interesting New APIs

SetMovieAudioOutputDeviceUID/ MediaSetAudioOutputDeviceUID

- Replaces the old `MediaSetSoundOutputComponent` API
- Old API will attempt to figure out the underlying device and do the right thing

Interesting New APIs

ExtractMovieAudio

- Pull-style API that extracts uncompressed audio from a movie
- It returns a CoreAudio-style AudioBufferList

Interesting New APIs

ExtractMovieAudio

- BeginMovieAudioExtraction
- EndMovieAudioExtraction
- SetMovieAudioExtractionProperty
- GetMovieAudioExtractionProperty/PropertyInfo
- ExtractMovieAudio

Demo:
ExtractMovieAudio

Exporting Audio

Brad Ford
QuickTime Engineering

Before QuickTime 6.4

- Exports limited to Sound Manager supported formats
- Problems with Standard Sound Compression (StdSound)
  - Requires displaying QuickTime's own dialog
  - Dialog offers strange choices for some compression formats:
    - Sample size choices for compressed audio
    - Uncompressed masquerading as compressed

New in QuickTime 6.4

- StdSound configures export to and from all our new playback formats
- StdSound works well with or without QuickTime's default dialog
  - Custom UI is easier to develop
  - New properties API makes this possible
- Default dialog is smart

Demo:
Exporting

What Application Changes Are Necessary?

- Export via ConvertMovieToFile/DataRef opts in automatically unless an open MovieExporter is passed in
- Apps that open a MovieExporter directly must opt in

```
QTSetComponentProperty(
     exporterInstance,
     kPropertyClass_MovieExporter,
     kMovieExporterPropertyID_Enable2003AudioFeatures,
     sizeof( Boolean ),
     &truevar);
```

If You have Opted in...

- Further changes only required if:
  - You call `MovieExportGetSettingsAsAtomContainer` AND parse the settings yourself
    - Because the settings are different

If You have Opted in...

- Further changes only required if:
  - You call `MovieExportGetSettingsAsAtomContainer` AND parse the settings yourself
    - Because the settings are different
  - You open StdSound to configure settings
    - Because you need to open the new StdSound
    - Because new StdSound has new APIs

The Properties Problem

- StdSound exposes standard properties as well as codec specific ones
- The existing API in StdSound cannot describe all the properties
  - With type safety
  - Without namespace collision
- New StdSound extends the StdSound APIs, so...

What Were Our Options?

Create generic set of property-related selectors

```
ComponentResult SCGetInfo(
    ComponentInstance           ci,
    OSType                      infoType,
    void *                      info);
...
```

- Drawbacks:
  - Properties identified solely by 4-char code—namespace collisions inevitable
  - Lacks type information and requires special knowledge from client

What Were Our Options?

New Get/Set APIs

- Create new Get/Set pairs and shim existing selectors

```
Fixed           SGGetSoundInputRate( SGChannel c);
ComponentResult SGSetSoundInputRate( SGChannel c,
                                     Fixed rate);
```

- Drawbacks:
  - Many new interfaces to learn
  - Does not address listenable properties
  - Some existing APIs require different data types

What Were Our Options?

Adopt a CoreAudio property model

```
ComponentResult
AudioCodecGetPropertyInfo(
    AudioCodec                  inCodec,
    AudioCodecPropertyID        inPropertyID,
    UInt32 *                    outSize,
    Boolean *                   outWritable);
...
```

- Drawbacks:
  - Each property class requires its own suite of calls, i.e. AudioCodec, AudioHardware, AudioDevice, AudioStream, etc.

QuickTime's Solution

A suite of Standard Component Property APIs

- New selectors are in the negative number range, like "Open", "Close", "Version", etc.
- Can be implemented by any component (including yours) as a standard means of exposing properties
- Any property may inform listeners of changes dynamically, via a user-supplied callback

QuickTime's Solution

A suite of Standard Component Property APIs

- Property type specified by a Property Class (4-char-code) and Property ID (another 4-char-code), for namespace protection
- Property data is explicitly typed and of known size (i.e. `kComponentDataTypeBoolean = 'bool'`)

QuickTime's Solution

A suite of Standard Component Property APIs

- "GetInfo" call tells you other characteristics:

```
kComponentPropertyFlagCanSetLater              = (1L << 0),
kComponentPropertyFlagCanSetNow                = (1L << 1),
kComponentPropertyFlagCanGetLater              = (1L << 2),
kComponentPropertyFlagCanGetNow                = (1L << 3),
kComponentPropertyFlagHasExtendedInfo          = (1L << 4),
kComponentPropertyFlagValueMustBeReleased      = (1L << 5),
kComponentPropertyFlagValueIsCFTypeRef         = (1L << 6),
kComponentPropertyFlagGetBufferMustBeInitialized = (1L << 7),
kComponentPropertyFlagWillNotifyListeners      = (1L << 8)
```

What Do They Look Like?

```
ComponentResult
CallComponentGetComponentPropertyInfo(
   ComponentInstance          inComponent,
   ComponentPropertyClass     inPropClass,
   ComponentPropertyID        inPropID,
   ComponentValueType *       outPropType,
   ByteCount *                outPropValueSize,
   UInt32 *                   outPropertyFlags);
```

StdSound Property Listener in Use

- Set up a listener proc…

```
    kSCAudioPropertyID_AvailableSampleRateList  = 'art#',
    /* indicates when sample rate list has been changed
       (e.g. compressor may have changed) */

void myListenerProc( ComponentInstance inComponent,
        ComponentPropertyClass inPropClass,
        ComponentPropertyID inPropID,
        void *inUserData)
{
    if (inPropClass == kPropertyClass_SCAudio && inPropID ==
       kSCAudioPropertyID_AvailableSampleRateList)
    {
        printf("Sample rate list has changed.\n");
        UpdateMyUI();
    }
}
```

StdSound Property Listener in Use

- In main:

```
ComponentPropertyListenerUPP myListenerUPP =
    NewComponentPropertyListenerUPP(myListenerProc);

err = CallComponentAddComponentPropertyListener(
        stdSoundci,                 // ComponentInstance
        kPropertyClass_SCAudio,     // ComponentPropertyClass
        kSCAudioPropertyID_AvailableSampleRateList,
                                    // kComponentPropertyID
        myListenerUPP,              // ComponentPropertyListenerUPP
        (void*)refcon);             // refcon

printf("Now listening for sample rate list changes\n");
CFRunLoopRun();
```

Important Note About Component Property APIs

Demo:
Audio Properties

DV Audio Changes

Tim Cherna
QuickTime Engineering

DV Audio in QuickTime < 6.3

- Audio from DV clips can be stored as PCM in its own track
  - Final Cut Pro and QuickTime Sequence Grabber do this
  - No problems with long-term A/V sync
- Audio track can index audio stored in DV stream
  - iMovie and QuickTime DV import do this
  - But, audio is represented as a fixed number of samples per DV frame!

DV Audio Problem

- Each frame is treated as a constant number of samples (e.g. 1602) in QuickTime's sample table
  - For playback, this is fine
  - For editing and export, a one-video frame drift accumulates about every 2 1/2 minutes
- Solution is to represent DV frames as variable duration
  - 'vdva' is new in QuickTime 6.3
  - Default for importing DV audio

DV Audio Reminders

- Sound tracks that index DV Audio the new way are:
  - `kVariableDurationDVAudioFormat` (‘vdva’)
- Old files still exist and have the old format:
  - `kDVAudioFormat` (‘dvca’)
- SoundDescriptions must have compressionID set to variableCompression

Audio/Video Startup Sync

Advanced QuickTime Programming Techniques

Audio Codecs

Eric M. Aldrich I
CoreAudio Engineering

What Is an Audio Codec?

- Handles the actual encoding/decoding of various audio formats
- Replaces sound manager components (scom/sdec)
- AAC, MP3, and IMA included in Panther

Writing an Audio Codec

- Use AudioCodecSDK
  - Provides framework needed to implement a codec
  - IMA 4:1 provided as an example
- Encoder and decoder can be implemented independently
- Use a push/pull model

Required Files

- ACCodec (.cpp and .h)
- ACBaseCodec (.cpp and .h)
- ACCodecDispatch (.cpp and .h)
- ACCodecDispatchTypes.h
- AudioCodec.h
- AudioCodecComponentGlue.c

Optional Files

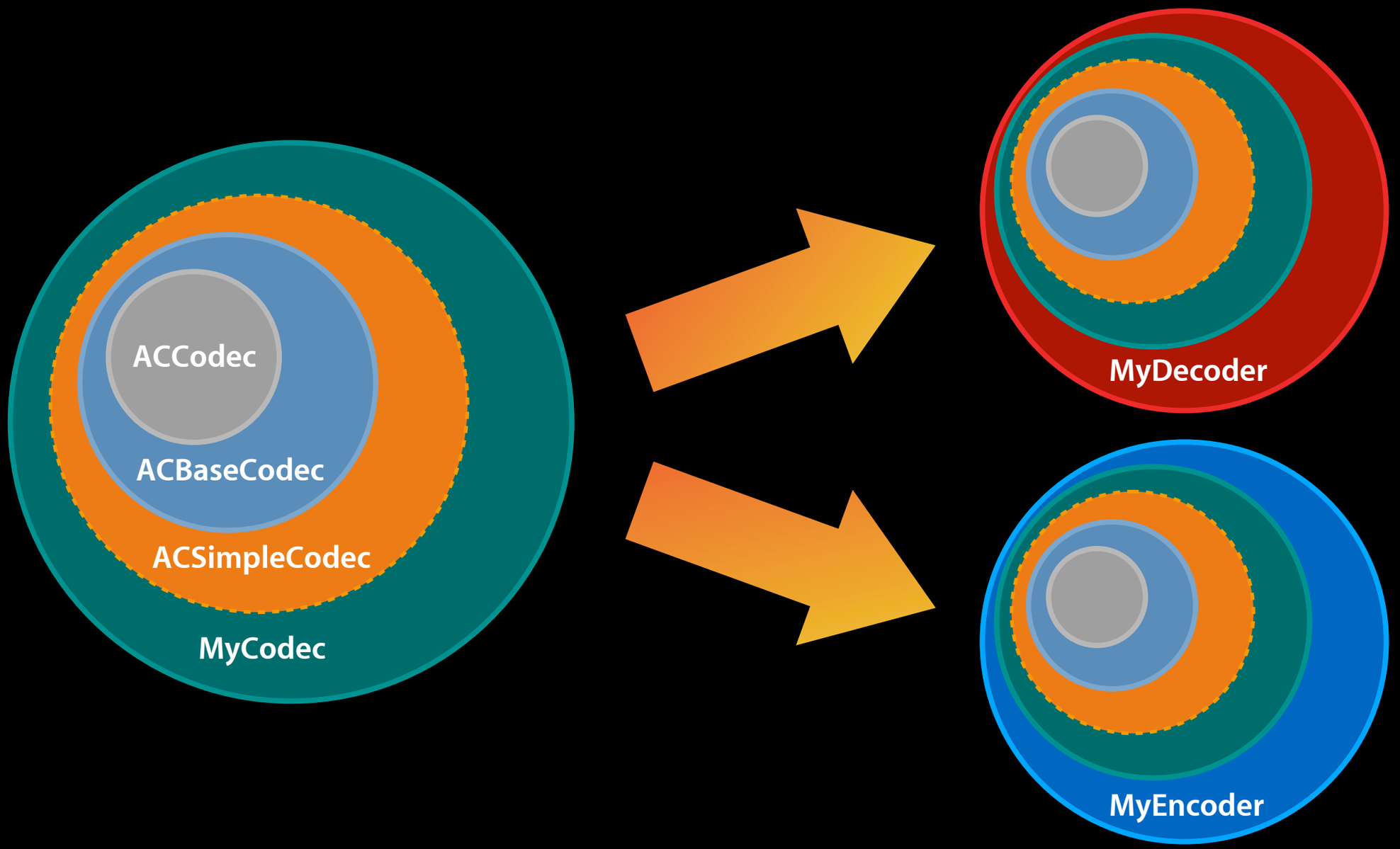
- ACSimpleCodec (.cpp and .h)
  - Provides built-in ring buffer
  - Implements AppendInputData method
  - Useful for basic codecs

Codec Specific Files

- Inherit directly from ACBaseCodec or ACSimpleCodec
- Primary methods to implement:
  - GetPropertyInfo, GetProperty, SetProperty
  - Initialize, Uninitialize
  - AppendInputData (if not using ACSimpleCodec)
  - ProduceOutputPackets

Odds and Ends

- Input/Output formats handled in constructor
- Encoder and decoder should be handled separately, but common functions can be handled in a base class



Dealing With Properties

- GetPropertyInfo tells you how much data a given property requires for Get/Set and whether the property can be set
- GetProperty
- SetProperty for a property is not always available if the codec is initialized

Initialize, Uninitialize

- Initialize locks down the configuration of the codec; Settings may not be changed
- Uninitialize simply removes this lock; settings may now be changed

AppendInputData

- Takes the input data and stores it in an internal buffer
- May work with or without AudioStreamPacketDescriptions
  - VBR formats should work with them
  - CBR formats do not need to

ProduceOutputPackets

- Does the actual encoding/decoding
- Clears the input buffer
- May work with or without AudioStreamPacketDescriptions

Backward Compatibility

- For old versions of QuickTime (before 6.4), we have Sound Manager Audio Components (SMAC)
- SMAC works as standard sdecs and scoms but “wraps” adecs and aencs
  - This model has already shipped since QuickTime 6.0
- May be implemented instead of full sdec/scom
- An example will soon be added to the AudioCodec SDK

Wrap Up

Roadmap

701 QuickTime State of the Union	The Presidio Tue., 9:00am
708 QuickTime APIs Update	Pacific Heights Tue., 5:00pm
405 Audio Formats	Russian Hill Tue., 5:00pm
716 Advanced QuickTime Programming Techniques	Pacific Heights Wed., 2:00pm
730 QuickTime Alternative Programming Environments	Pacific Heights Fri., 10:30pm
727 QuickTime Streaming Server Programming	The Mission Thu., 2:00pm

Roadmap (Cont.)

FF011 QuickTime Feedback Forum	**Haight/Ashbury** **Thu., 3:30pm**

Who to Contact

Guillermo Ortiz
QuickTime Technology Manager
quicktimeman@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Documentation

General Information

http://www.apple.com/quicktime

Developer Information

http://developer.apple.com/documentation

QuickTime Information

http://developer.apple.com/QuickTime

Do Not Miss!

Lab: QuickTime Content Creation

Meet with QuickTime content experts for daily content specific topics that are of special interest to you. Bring your own media and Powerbook. No appointment is needed—first-come, first-served.

Sacramento Room

Lab: QuickTime Development

Meet with Apple's QuickTime engineers. Each day will emphasize a specific QuickTime development topic. Bring your own code and Powerbook. No appointment is needed—first-come, first-served.

Sacramento Room

Lab Hours

Monday	12:00pm to 6:00pm
Tuesday	9:00am to 6:00pm
Wednesday	9:00am to 6:00pm
Thursday	9:00am to 4:30pm
Friday	9:00am to 6:00pm

Q&A

Guillermo Ortiz
QuickTime Technology Manager
quicktimeman@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference